Statistical Learning of Arbitrary Computable Classifiers

David Soloveichik*
California Institute of Technology
MC 136-93
Pasadena, CA 91125
dsolov@caltech.edu

Abstract

Statistical learning theory chiefly studies restricted hypothesis classes, particularly those with finite Vapnik-Chervonenkis (VC) dimension. The fundamental quantity of interest is the sample complexity: the number of samples required to learn to a specified level of accuracy. Here we consider learning over the set of all computable labeling functions. Since the VC-dimension is infinite and a priori (uniform) bounds on the number of samples are impossible, we let the learning algorithm decide when it has seen sufficient samples to have learned. We first show that learning in this setting is indeed possible, and develop a learning algorithm. We then show, however, that bounding sample complexity independently of the distribution is impossible. Notably, this impossibility is entirely due to the requirement that the learning algorithm be computable, and not due to the statistical nature of the problem.

1 Introduction

Suppose we are trying to learn a difficult classification problem: for example determining whether the given image contains a human face, or whether the MRI image shows a malignant tumor, etc. We may first try to train a simple model such as a small neural network. If that fails, we may move on to other, potentially more complex, methods of classification such as support vector machines with different kernels, techniques to apply certain transformations to the data first, etc. Conventional statistical learning theory attempts to bound the number of samples needed to learn to a specified level of accuracy for each of the above models (e.g. neural networks, support vector machines). Specifically, it is enough to bound the VC-dimension of the learning model to determine the number of samples to use [VC71, BEHW89]. However, if we allow ourselves to change the model, then the VC-dimension of the overall learning algorithm is not finite, and much of statistical learning theory does not directly apply.

Accepting that much of the time the complexity of the model cannot be a priori bounded, Structural Risk Minimization [Vap98] explicitly considers a hierarchy of increasingly complex models. An alternative approach, and one we follow in this paper, is simply to consider a single learning model that includes all possible classification methods.

We consider the unrestricted learning model consisting of all computable classifiers. Since the VC-dimension is clearly infinite, there are no uniform bounds (independent of the distribution and the target concept) on the number of samples needed to learn accurately [BEHW89]. Yet we still want to guarantee a desired level of accuracy. Rather than deciding on the number of samples a priori, it is natural to allow the learning algorithm to decide when it has seen sufficiently many labeled samples based on the training samples seen up to now and their labels. Since the above learning model includes any practical classification scheme, we term it universal (PAC-) learning.

We first show that there is a computable learning algorithm in our universal setting. Then, in order to obtain bounds on the number of training samples that would be needed, we consider measuring sample complexity of the learning algorithm as a function of the unknown correct labeling function (i.e. target concept). Although the correct labeling is unknown, this sample complexity measure could be used to compare learning algorithms speculatively: "if the target labeling were such and such, learning algorithm A requires fewer samples than learning algorithm B". By asking what is the largest sample size needed assuming the target labeling function is in a certain class, we could compare the sample complexity of the universal learner to a learner over the restricted class (e.g. with finite VC-dimension).

However, we prove that it is impossible to bound the sample complexity of any *computable* universal learning algorithm, even as a function of the target concept. Depending on the distribution, any such bound will be exceeded with arbitrarily high probability. The impossibility of a distribution-independent bound is entirely due to the computability requirement. Indeed we show there is an uncomputable learning procedure for which we bound the number of samples queried as a function of the unknown target concept, independently of the distribution.

Our results imply that computable learning algorithms in the universal setting must "waste samples" in the sense of requiring more samples than is necessary for statistical reasons alone.

*I thank Erik Winfree and Matthew Cook for discussions and invaluable support.

2 Relation to Previous Work

There is comparatively little work in statistical learning theory on learning arbitrary computable classifiers compared to the volume of research on learning in more restricted settings. Computational learning theory (aka PAC-learning) requires learning algorithms to be efficient in the sense of running in polynomial time of certain parameters [Val84, KV94]. That work generally restricts learning to very limited concept/hypothesis spaces such as perceptrons, DNF expressions, limited-weight neural networks, etc. The purely statistical learning theory paradigm ignores issues of computability [VC71, Vap98]. Work on learning arbitrary computable functions is mostly in the "learning in the limit" paradigm [Gol67, Ang88], in which the goal of learning is to eventually converge to the perfectly correct hypothesis as opposed to approximating it with an approximately correct hypothesis.

The idea of allowing the learner to ask for a varying number of training samples based on the ones previously seen was studied before in statistical learning theory [LMR88, BI94]. Linial et al [LMR88] called this model "dynamic sampling" and showed that dynamic sampling allows learning with a hypothesis space of infinite VC-dimension if all hypotheses can be enumerated. This is essentially Theorem 4 of our paper. However, the hypothesis space of all computable functions cannot be enumerated by any algorithm, and thus these results do not directly imply the existence of a learning algorithm in our setting.

Our proof technique for establishing positive results (Theorem 2) is parallel evaluation of all hypotheses, and is based on Levin's universal search [Lev73]. In learning theory, Levin's universal search was previously used by Goldreich and Ron [GR97] to evaluate all learning algorithms in parallel and obtain an algorithm with asymptotically optimal computation time.

The main negative result of this paper is showing the absence of distribution independent bounds on sample complexity for computable universal learning algorithms (Theorem 5). Recently Ryabko [Rya05] considered learning arbitrary computable classifiers, albeit in a setting where the number of samples for the learning algorithm is externally chosen. He demonstrated a computational difficulty in determining the number of samples needed: it grows faster than any computable function of the length of the target concept. In contrast, we prove that distribution-independent bounds do not exist altogether for computable learning algorithms in our setting.

3 Definitions

The *sample space* X is the universe of possible points over which learning occurs. Here we will largely suppose the sample space X is the set of all finite binary strings $\{0,1\}^*$. A *concept space* C and *hypothesis space* H are sets of boolean-valued functions over X, which are said to *label* points $x \in X$ as $0/1$. The concept space C is the set of all possible labeling functions that our learning algorithm may be asked to learn from. In each learning scenario, there is some unknown *target concept* $c \in C$ that represents the desired way of labeling points. There is also an unknown *sample distribution* D over X. The learning algorithm chooses a *hypothesis* $h \in H$ based on iid samples drawn from D and labeled according to the target concept c. Since we cannot hope to distinguish between a hypothesis that is always correct and one that is correct most of the time, we adopt the "probably approximately correct" [Val84] goal of producing with high probability $(1-\delta)$ a hypothesis h such that the probability over $x \sim D$ that $h(x) \neq c(x)$ is small (ε).

Here we will mostly consider the concept space C to be the set of all total recursive functions $X \to \{0,1\}$. We say that this is a universal learning setting because C includes any practical classification scheme. We will mostly consider the hypothesis space to be the set of all partial recursive functions $X \to \{0,1,\perp\}$, where $\perp$ indicates failure to halt. From PAC learning it is known that sometimes it helps to use different concept and hypothesis classes, if one desires the learning algorithm to be efficient [PV88]. In a related way, allowing our algorithm to output a partial recursive function that may not halt on all inputs seems to permit learning (e.g. Theorem 2). Abusing notation, $c \in C$ or $h \in H$ will refer to either the function or to a representation of that function as a program. Similarly C and H will refer to the sets of functions or to the sets of representations of the corresponding functions. We assume all programs are written in some fixed alphabet and are interpreted by some fixed universal Turing machine. If h is a partial recursive function and $h(x) = \perp$ then by convention $h(x) \neq h'(x)$ for any partial recursive function h' (even if $h'(x) = \perp$ also).

We can now define what we mean by a learning algorithm:

Definition 1 *Algorithm A is a* learning algorithm *over sample space X, concept space C, and hypothesis space H if:*

- *(syntactic requirements) A takes two inputs $\delta \in (0,1)$ and $\varepsilon \in (0,1/2)$, queries an oracle for pairs in $X \times \{0,1\}$, and if A halts it outputs a hypothesis $h \in H$.*
- *(semantic requirements) For any δ, ε, for any concept $c \in C$, and distribution D over X, if the oracle returns pairs $(x, c(x))$ for x drawn iid from D, then A always halts, and with probability at least $1-\delta$ outputs a hypothesis h such that $\Pr_{x\sim D}[h(x) \neq c(x)] < \varepsilon$.*

The always halting requirement seems a nice property of the learning algorithm and indeed the learning algorithm we develop (Theorem 2) will halt for any concept and sequence of samples. However, relaxing this requirement to allow a non-zero probability that the learning algorithm queries the oracle for infinitely many samples does not change our negative results (Theorem 5), as long as a finite number of oracle calls implies halting.

The fundamental notion in statistical learning theory is that of sample complexity. Since the VC-dimension of our hypothesis space is infinite, there is no *uniform bound* $m(\delta, \varepsilon)$ on the number of samples needed to learn to the δ, ε level of accuracy. We will consider the question of whether for a given learning algorithm there is a *distribution-independent bound* $m(c, \delta, \varepsilon)$ on the number of samples queried from the oracle where $c \in C$ is the target hypothesis. In other words the bound is allowed to depend on the target concept c but not on the sample distribution D. Such a

bound may be satisfied with certainty, or satisfied with high probability over the learning samples.

4 Results

We first show that there is a computable learning algorithm in our setting.

Theorem 2 *There is a learning algorithm over sample space X of all finite binary strings, hypothesis space H of all partial recursive functions, and concept space C of all total recursive functions.*

In order to prove this theorem we need the following lemma. Results equivalent to this lemma can be found in [LMR88].

Lemma 3 *Let X be any sample space and D be any distribution over X. Fix any function $c : X \to \{0,1\}$. Suppose hypothesis space H is countable, and let $h_1, h_2, \ldots$ be some ordering of H. For any δ, ε, let $m(i) = \lceil (2\ln i + \ln(1/\delta) + \ln(\pi^2/6))/\varepsilon \rceil$. Suppose $x_1, x_2, \ldots$ is an infinite sequence of iid samples drawn from D. Then the probability that there exists $h_i \in H$ such that $\Pr_{x\sim D}[h_i(x) \neq c(x)] > \varepsilon$, but h_i agrees with c on $x_1, x_2, \ldots, x_{m(i)}$, is less than δ.*

Proof: The probability that a particular h_i with error probability $\Pr_{x\sim D}[h_i(x) \neq c(x)] > \varepsilon$ gets $m(i)$ i.i.d. instances drawn from D correct is less than $(1-\varepsilon)^{m(i)} \leq e^{-m(i)\varepsilon} \leq (6/\pi^2)(\delta/i^2)$. By the union bound, the probability that *any* h_i with error probability greater than ε gets $m(i)$ instances correct is less than $\sum_{i=1}^{\infty}(6/\pi^2)(\delta/i^2) = \delta$. ■

Proof of Theorem 2: Let $h_1, h_2, \ldots$ be a recursive enumeration of H (for example in lexicographic order). For the given δ, ε, let $m(i)$ be defined as in Lemma 3. The learning algorithm computes infinitely many threads $1, 2, \ldots$ running in parallel. This can be done by a standard dovetailing technique. (For example use the following schedule: for $k = 1$ to infinity, for $i = 1$ to k, perform step $k - i + 1$ of thread i.) Thread i sequentially checks whether $h_i(x_1) = c(x_1)$, $h_i(x_2) = c(x_2)$, $\ldots$, $h_i(x_{m(i)}) = c(x_{m(i)})$, exiting if a check fails. If all $m(i)$ checks pass, thread i terminates and outputs h_i. The learning algorithm queries the oracle as necessary for new learning samples and their labeling. The overall algorithm terminates as soon as some thread outputs an h_i, and outputs this hypothesis. By Lemma 3, with probability at least $1-\delta$, this h_i has error probability less than ε. Further, since $C \subset H$, the learning algorithm will always terminate. ■

Note that it seems necessary to expand the hypothesis space to include all partial recursive functions because the concept space of total recursive functions does not have a recursive enumeration (it is uncomputable whether a given program is total recursive or not).

We will see in Theorem 5 that there is no bound $m(c, \delta, \varepsilon)$ on the number of samples queried by any computable learning algorithm in our setting. Let us obtain some intuition for why that is true for the above learning algorithm. Then we will contrast this to the case of an uncomputable learning algorithm.

In essence, we can make the above learning algorithm query for more samples than is necessary for statistical reasons alone. Intuitively, suppose that an h_{i^*} coming early in the ordering is always correct but takes a very long time to compute. The learning algorithm cannot wait for this h_{i^*} to finish, because it does not know that any particular h_i will ever halt. At some point it has to start testing h_i's that come later in the ordering and that have larger $m(i)$'s. Testing these requires more learning samples than $m(i^*)$.

If we can know which h_i's are safe to skip over since they don't halt, and for which h_i's we should wait, then the above problem is solved. Indeed, the following theorem shows that there is no statistical reason why a distribution-independent bound $m(c, \delta, \varepsilon)$ is impossible. The theorem presents a well defined method of learning (albeit an uncomputable one) for which there exists such a bound, and this bound is satisfied with certainty. Below, the halting oracle gives $0/1$ answers to questions of the form (h, x) where $h \in H, x \in X$ such that a 1 answer indicates that $h(x)$ halts and a 0 answer indicates it does not; the answers are clearly uncomputable.

Theorem 4 *If a learning algorithm is allowed to query the halting oracle, then there is a learning algorithm over sample space X of all finite binary strings, hypothesis space H of all partial recursive functions, and concept space C of all total recursive functions, and a function $m : C \times (0,1) \times (0,1/2) \to \mathbb{N}$, such that for any approximation parameters δ, ε, any target concept $c \in C$, and any distribution D over X, the learning algorithm uses at most $m(c, \delta, \varepsilon)$ training samples.*

Proof: Rather than dovetailing as is done for the computable learning algorithm (Theorem 2), we can sequentially test every h_i on samples $x_1, \ldots, x_{m(i)}$ because we can determine whether h_i halts on a given input. Since $c = h_{i^*}$ for some $h_{i^*} \in H$, the hypothesis h_i we output will always satisfy $i < i^*$, and therefore we will require at most $m(i^*) = \lceil (2\ln(i^*) + \ln(1/\delta) + \ln(\pi^2/6))/\varepsilon \rceil$ samples. ■

We now show that for any *computable* learning algorithm, and any possible sample bound $m(c, \delta, \varepsilon)$, there is a target concept c and a sample distribution such that this sample bound is violated with high probability. The probability of violation can be made arbitrarily close to $1-2(\delta+(1-\delta)\varepsilon)$ (which approaches 1 as $\delta, \varepsilon \to 0$). In fact this theorem is stronger: it shows that given a learning algorithm, without varying the target concept, but just by varying the distribution it is possible to make the algorithm ask for arbitrarily many learning samples with high probability.

Theorem 5 *For any learning algorithm over sample space X of all finite binary strings, hypothesis space H of all partial recursive functions, and concept space C of all total recursive functions, there is a target concept $c \in C$, such that for any approximation parameters δ, ε, for any $\rho < 1 - 2(\delta + (1-\delta)\varepsilon)$, and for any sample bound $m \in \mathbb{N}$ there is a distribution D over X, such that the learning algorithm uses more than m training samples with probability at least ρ.*

The key difference between a computable and an uncomputable learning algorithm, is that a concept can simulate

a computable one. By simulating the learning algorithm, a concept can choose to behave in way that is bad for the learning algorithm's sample complexity.

To prove the above theorem, we will first need the following lemma. The lemma essentially shows a situation such that any learning algorithm according to our definition must query for more than m learning samples with high probability when the target concept is chosen adversarily. The lemma is true even without requiring the learning algorithm to be computable. Note that the lemma does not directly imply the theorem above, even in its weaker form, because in order to increase the number of learning samples that are likely queried by the learning algorithm, we have to change the target concept. Since $m(c, \delta, \varepsilon)$ is a function of c, there is no guarantee that the bound doesn't become larger as well.

Lemma 6 *Let X be a set of d points, and let C be the set of all labelings of X. Let D be a uniform distribution over X. Suppose A is a learning algorithm over sample space X, concept and hypothesis space C. For any accuracy parameters δ, ε and any $m < d$, there is a concept $c \in C$ such that when the oracle draws from D labeled according to c the probability that A samples more than m points is at least $1 - \frac{2d(\delta+(1-\delta)\varepsilon)}{d-m}$.*

Proof: We use the probabilistic method to find a particularly bad concept c^*. Suppose we do not start with a fixed target concept c, but draw it uniformly from C. In other words, c is determined by values $\{c(x)\}_{x \in X}$ drawn uniformly from $\{0, 1\}$. Given some $x_1, \ldots, x_m$, $c(x_1), \ldots, c(x_m)$, and $x \notin \{x_1, \ldots, x_m\}$, the value of $c(x)$ is a fair coin flip. Thus if on $x_1, \ldots, x_m$ labeled by $c(x_1), \ldots, c(x_m)$, A outputs a hypothesis without asking for more samples, then the hypothesis is incorrect on x with probability $1/2$. If we now let x vary, the probability that the hypothesis is incorrect on x is at least $(1/2)(d - m)/d$ since there are at least $d - m$ points not in $x_1, \ldots, x_m$. Now suppose for any c the probability that A samples more than m points is at most ρ. Then the unconditional probability that the hypothesis output by A is incorrect on a random sample point is at least $(1 - \rho)(1/2)(d - m)/d$. This implies that there is a concept $c^* \in C$ such that the probability that the hypothesis output by A is incorrect on a random sample point is at least $(1 - \rho)(1/2)(d - m)/d$.

Since A is a learning algorithm, when we use c^* to label the training points, and use accuracy parameters δ, ε, the probability that the hypothesis produced by A has error probability greater than ε is at most δ. If we make the worst case assumption that whenever the error probability of the hypothesis is larger than ε it is exactly 1, and otherwise the error probability is exactly ε, then the probability that the hypothesis output by A is incorrect on a random sample point is at most $\delta \cdot 1 + (1 - \delta)\varepsilon$. Thus $(1 - \rho)(1/2)(d - m)/d \leq \delta + (1 - \delta)\varepsilon$, implying that $\rho \geq 1 - \frac{2d(\delta+(1-\delta)\varepsilon)}{d-m}$. ■

Now in order to prove Theorem 5, we essentially show that there is some fixed concept c^* that behaves as the bad c's in arbitrary instances of Lemma 6.

Proof of Theorem 5: Consider the following program $P : \{0, 1\}^* \to \{0, 1\}$. First it interprets the given string $x \in \{0, 1\}^*$ as a tuple $\langle \delta, \varepsilon, m, d, i \rangle$ for $\delta \in (0, 1)$, $\varepsilon \in (0, 1/2)$ and $m, d, i \in \mathbb{N}$ using some fixed one-to-one encoding of such tuples as binary strings. If x cannot be decoded appropriately, or if $i > d$ then P returns 0. Otherwise, for these $\delta, \varepsilon, m, d$, let $\hat{X} \subset \{0, 1\}^*$ be the set of d strings which are interpreted as $\{\langle \delta, \varepsilon, m, d, 1 \rangle, \ldots, \langle \delta, \varepsilon, m, d, d \rangle\}$, and let $\hat{D}$ be a uniform distribution over $\hat{X}$ and 0 elsewhere. Let $\hat{C}$ be the set of all possible labelings of $\hat{X}$. For each labeling $\hat{c} \in \hat{C}$, program P computes the probability $\rho_{\hat{c}}$ that A given accuracy parameters δ, ε, queries for more than m sample points if points are drawn from $\hat{D}$ labeled according to $\hat{c}$. For each $\hat{c}$, this requires simulating A for at most d^m different sequences of sample points. Let $\hat{c}^* = \operatorname{argmax}_{\hat{c} \in \hat{C}}\{\rho_{\hat{c}}\}$, breaking ties in some fixed way. Finally P outputs $\hat{c}^*(x)$.

Observe that P is total recursive since A spends a finite time on any finite sequence of sample points. (This is a weaker condition than the always halting requirement of our definition of a learning algorithm.) Thus P is some $c^* \in C$. Further, for any $\delta, \varepsilon, m, d$, on all points $\langle \delta, \varepsilon, m, d, i \rangle$ for $i \leq d$, P finds the same $\hat{c}^*$, and thus on these points c^* acts like this $\hat{c}^*$. By Lemma 6, if $m < d$ then this $\hat{c}^*$ has the property that $\rho_{\hat{c}^*} \geq 1 - \frac{2d(\delta+(1-\delta)\varepsilon)}{d-m}$. Therefore, if A is given accuracy parameters δ, ε, the target concept is c^*, and the distribution D is uniform over $\{\langle \delta, \varepsilon, m, d, 1 \rangle, \ldots, \langle \delta, \varepsilon, m, d, d \rangle\}$ for some $d \in \mathbb{N}$ such that $m < d$, then the probability that A requests more than m samples is at least $1 - \frac{2d(\delta+(1-\delta)\varepsilon)}{d-m}$. Since we can choose D such that d is large enough, we obtain the desired result. ■

5 Conclusion

We have shown that learning arbitrary computable classifiers is possible in the statistical learning paradigm. However for any computable learning algorithm, the number of samples required to learn to a desired level of accuracy may become arbitrarily large depending on the sample distribution. This is in contrast to uncomputable learning methods in the same universal setting whose sample complexity can be bounded independently of the distribution.

Our results mean that there is a big price in terms of sample complexity to be paid for the combination of universality and computability of the learner. Specifically, by tweaking the distribution we can make a computable universal learner arbitrarily worse than a restricted learning algorithm on a finite VC-dimensional hypothesis space, or even an uncomputable universal learner.

While we have presented a single computable learning algorithm in our universal setting, one would like to develop a measure that would allow different learning algorithms to be compared to each other in terms of sample complexity. We have seen that sample complexity $m(c, \delta, \varepsilon)$ is not such a measure; is there a viable alternative?

Finally, we have ignored computation time in our analysis. As such, our learning algorithm is not likely to have practical significance. Integrating running time into the theory presented would be a critical extension.

References

[Ang88] D. Angluin. Identifying languages from stochastic examples. Technical report, Yale University, Department of Computer Science, 1988.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[BI94] G. M. Benedek and A. Itai. Nonuniform learnability. *Journal of Computer and System Sciences*, pages 311–323, 1994.

[Gol67] E. M. Gold. Language Identification in the Limit. *Information and Control*, 10:447–474, 1967.

[GR97] O. Goldreich and D. Ron. On universal learning algorithms. *Information Processing Letters*, 63(3):131–136, 1997.

[KV94] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[Lev73] L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

[LMR88] N. Linial, Y. Mansour, and R. L. Rivest. Results on learnability and the Vapnik-Chervonenkis dimension. *29th Annual Symposium on Foundations of Computer Science*, pages 120–129, 1988.

[PV88] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

[Rya05] D. Ryabko. On Computability of Pattern Recognition Problems. In *Proceedings of the 16th International Conference on Algorithmic Learning Theory*, pages 148–156. Springer, 2005.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[Vap98] V. N. Vapnik. *Statistical learning theory*. Wiley New York, 1998.

[VC71] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.